Introduction to Programming in C

(IS-FEE-10061S)

Białystok University of Technology
Faculty of Electrical Engineering

Academic year 2023/2024

Workshop no. 04 (21.03.2024)

Jarosław Forenc, PhD

Topics

- The for loop
- The increment (++) and decrement (--) operators
- The while loop
- The do … while loop

Example: sum of numbers

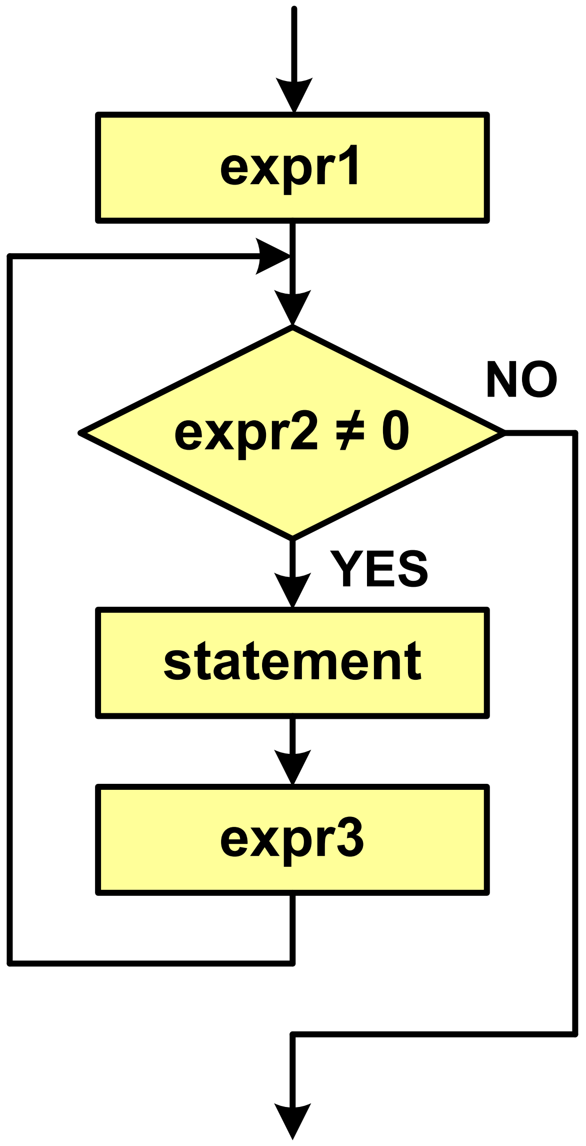
```
#include <stdio.h>

int main(void)
{
    int sum, i;

    sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("The sum is: %d\n",sum);

    sum = 0;
    for (i=1; i<=100; i=i+1)
        sum = sum + i;

    printf("The sum is: %d\n",sum);

    return 0;
}
```

```
The sum is: 55
The sum is: 5050
```

The for loop

```
for (expr1; expr2; expr3)
    statement
```

- expr1, expr2, expr3 - any expression in the C language
- Statement:
  - single - one statement terminated by a semicolon
  - compound - one or more statements enclosed in braces



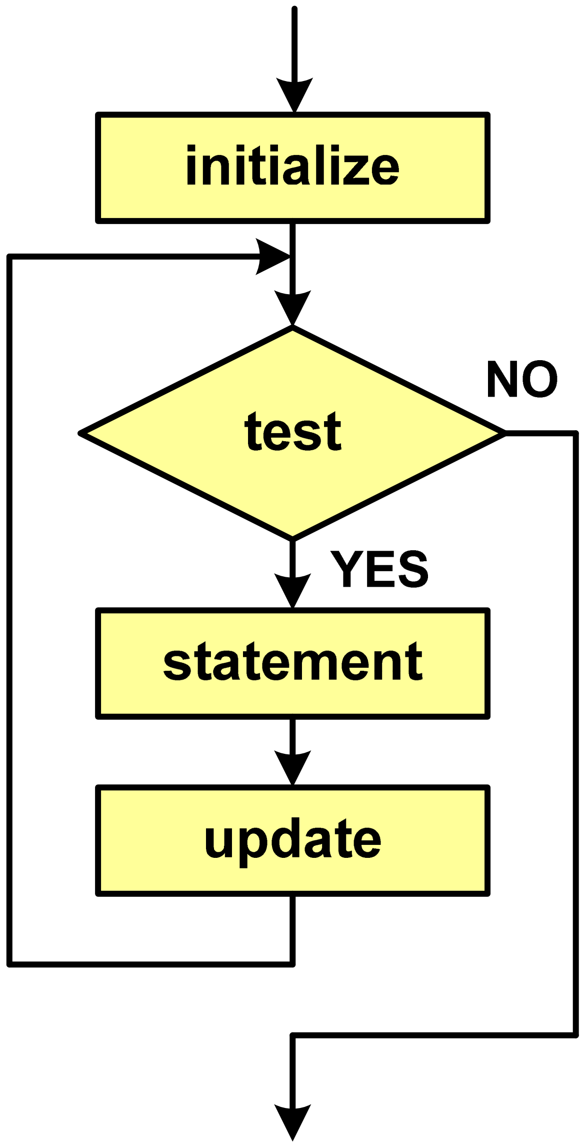
The for loop

- The most common form of the for loop

```
int i;
for (i = 0; i < 10; i = i + 1)
    statement;
```

- The instruction will be executed 10 times (for i = 0, 1, 2, … 9)
- Functions of expressions

```
for (initialize; test; update)
    statement
```



Example: printing the text 5 times

```
#include <stdio.h>

int main(void)
{
    int i;

    for (i=0; i<5; i=i+1)
        printf("Programming is not difficult\n");

    return 0;
}
```

```
Programming is not difficult
Programming is not difficult
Programming is not difficult
Programming is not difficult
Programming is not difficult
```

The for loop (examples)

```
for (i=0; i<10; i++)
    printf("%d ",i);
```

```
0 1 2 3 4 5 6 7 8 9
```

```
for (i=0; i<10; i++)
    printf("%d ",i+1);
```

```
1 2 3 4 5 6 7 8 9 10
```

```
for (i=1; i<=10; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 10
```

The for loop (examples)

```
for (i=1; i<10; i=i+2)
    printf("%d ",i);
```

```
1 3 5 7 9
```

```
for (i=10; i>0; i--)
    printf("%d ",i);
```

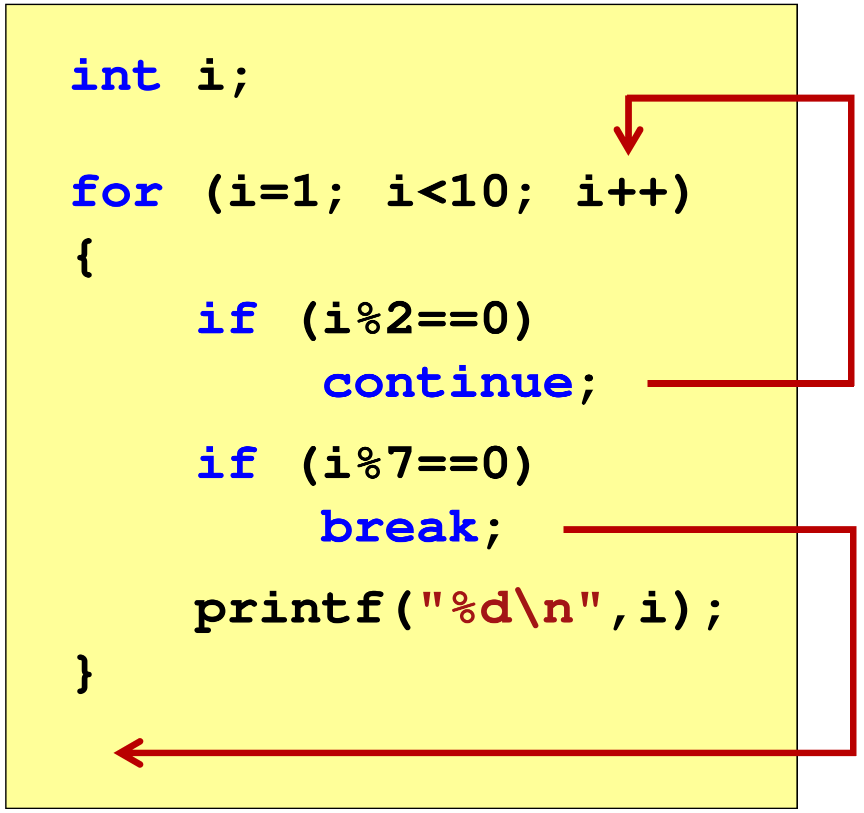
```
10 9 8 7 6 5 4 3 2 1
```

```
for (i=-9; i<=9; i=i+3)
    printf("%d ",i);
```

```
-9 -6 -3 0 3 6 9
```

The for loop (break, continue)

- The following statements can be used in a for loop: break, continue



```c
int i;

for (i=1; i<10; i++)
{
    if (i%2==0)
        continue;
    if (i%7==0)
        break;
    printf("%d\n",i);
}
```

```
1 3 5
```

- continue terminates the current iteration and proceeds to evaluate expr3
- break terminates the loop execution

The for loop (most common mistakes)

- Putting a semicolon at the end of the for loop

```
int i;
for (i=0; i<10; i++);
printf("%d ",i);
```

```
10
```

- Comma instead of semicolons between expressions

```
int i;
for (i=0, i<10, i++)
    printf("%d ",i);
```

Compilation error!

The for loop (most common mistakes)

- Incorrect condition - no execution of the instruction

```
int i;
for (i=0; i>10; i++)
    printf("%d ",i);
```

- Incorrect condition - infinite loop

```
int i;
for (i=1; i>0; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 ...
```

Nested loops

- There may be another for loop as a statement in the for loop

```
int i, j;
for (i=1; i<=3; i++)           // outer loop
    for (j=1; j<=2; j++)       // inner loop
        printf("i: %d   j: %d\n",i,j);
```

```
i: 1   j: 1
i: 1   j: 2
i: 2   j: 1
i: 2   j: 2
i: 3   j: 1
i: 3   j: 2
```

- a common use for nested loops is to display data in rows and columns

The increment operator (++)

- The unary ++ operator increments (increases) the value of its operand by 1 (not allowed for expressions)
- The ++ operator can be either a prefix or a suffix

Notation	Mode	Meaning
`++x`	the prefix mode	the variable is changed before its value is used
`x++`	the postfix mode	the variable is changed after its value is used

The increment operator (++)

- Example

```
int x = 1, y;
y = 2 * ++x;
```

```
int x = 1, y;
y = 2 * x++;
```

- The order of operations

```
++x                x = 2
2 * ++x            2 * 2
y = 2 * ++x        y = 4
```

```
2 * x              2 * 1
y = 2 * x          y = 2
x++                x = 2
```

- Variable values

```
x = 2     y = 4
```

```
x = 2     y = 2
```

The increment operator (++)

- The position of the ++ operator does not matter in the case of statements like:

```
x++;
++x;
```

equivalent

```
x = x + 1;
```

- Do not use the ++ operator on a variable that appears more than once in an expression

```
x = x++;
x = ++x;
```

- According to the C language standard, the result of the above statements is undefined

The decrement operator (--)

- The unary -- operator decrements (decreases) the value of its operand by 1 (not allowed for expressions)
- The -- operator can be either a prefix or a suffix

Notation	Mode	Meaning
`--x`	the prefix mode	the variable is changed before its value is used
`x--`	the postfix mode	the variable is changed after its value is used

Example: square root

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Enter number: ");
    scanf("%f",&x);

    if (x>=0)
    {
        y = sqrt(x);
        printf("Square root: %f\n",y);
    }
    else
        printf("Error! Negative number\n");

    return 0;
}
```

```
Enter number: -3
Error! Negative number
```

```
Enter number: 3
Square root: 1.732051
```

Example: square root (the while loop)

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Enter number: ");
    scanf("%f",&x);
    while (x<0)
    {
        printf("Error! Negative number\n\n");
        printf("Enter number : ");
        scanf("%f",&x);
    }
    y = sqrt(x);
    printf("Square root: %f\n",y);

    return 0;
}
```

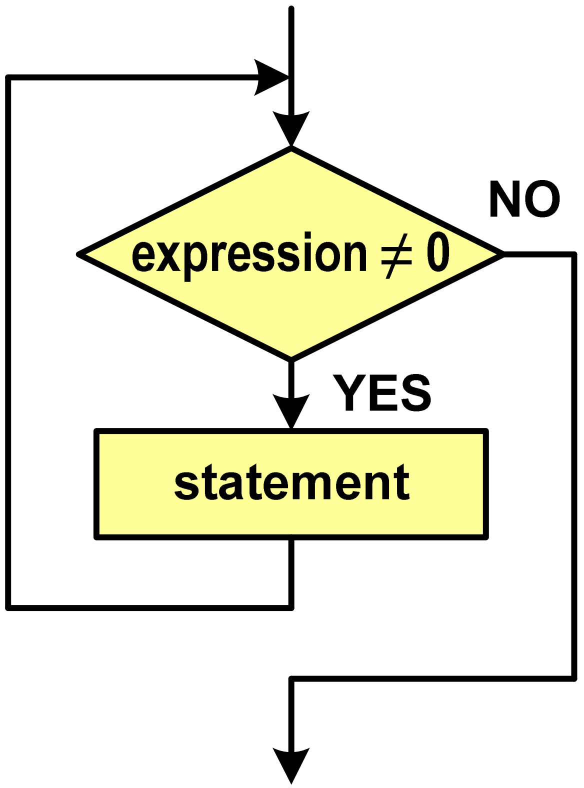
```
Enter number: -3
Error! Negative number

Enter number: -5
Error! Negative number

Enter number: 3
Square root: 1.732051
```

The while loop

```
while (expression)
      statement
```

- "as long as the expression in parentheses is true, execute the statement"

- Expression:
  - true - when its value is different from zero (nonzero)
  - false - when its value is zero
- As an expression, a logical expression is most often used



The while loop

```
while (expression)
    statement
```

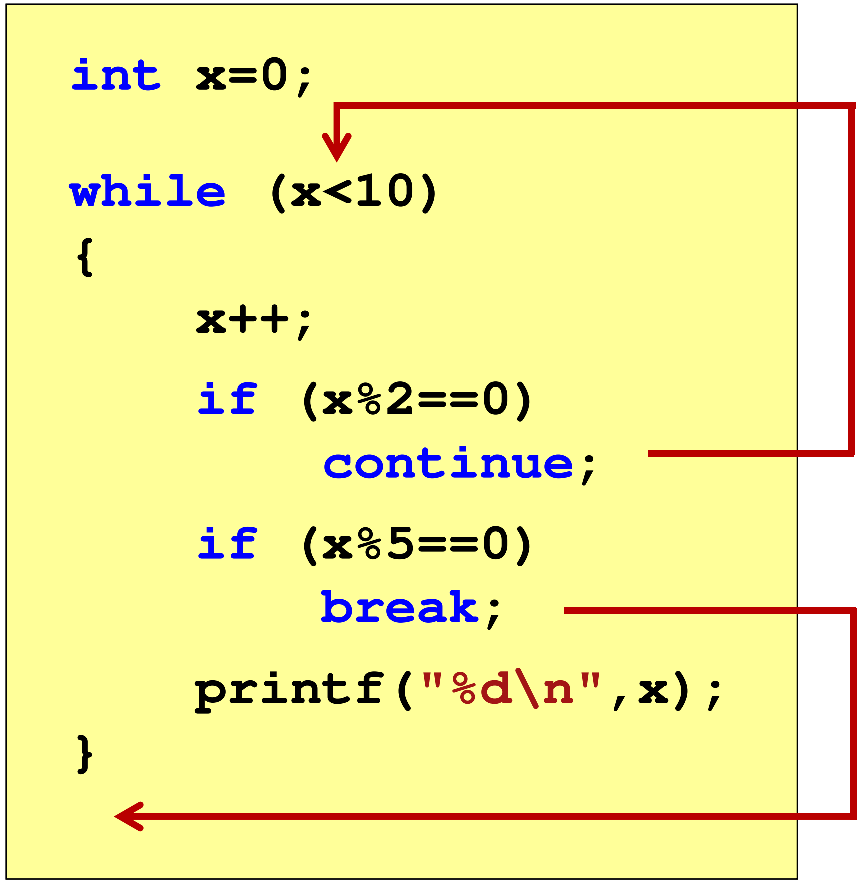
- Statement:
  - single - one statement terminated by a semicolon
  - compound - one or more statements enclosed in braces

```
int x = 10;
while (x>0)
    x = x - 1;
```

```
int x = 10;
while (x>0)
{
    printf("%d\n",x);
    x = x - 1;
}
```

The while loop (break, continue)

- break and continue are jump statements



```c
int x=0;

while (x<10)
{
    x++;
    if (x%2==0)
        continue;
    if (x%5==0)
        break;
    printf("%d\n",x);
}
```

- continue terminates the current iteration
- break terminates the loop execution

The while loop (most common mistakes)

- Putting a semicolon after the expression in brackets causes an infinite loop - the program stops on the loop

```
int x = 10;
while (x>0);
    printf("%d ",x--);
```

- Lack of updating the variable also causes an infinite loop - the program prints the same value many times

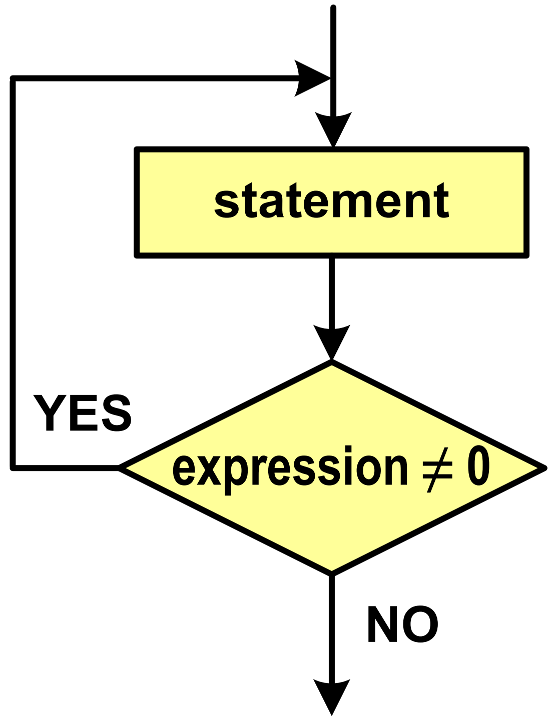
```
int x = 10;
while (x>0)
    printf("%d ",x);
```

```
10 10 10 10 10 ...
```

The do while loop

```
do
    statement
while (expression);
```

- "execute the statement as long as the expression in parentheses is true"



- Expression:
  - true - when its value is different from zero (nonzero)
  - false - when its value is zero

The do while loop

```
do
    statement
while (expression);
```

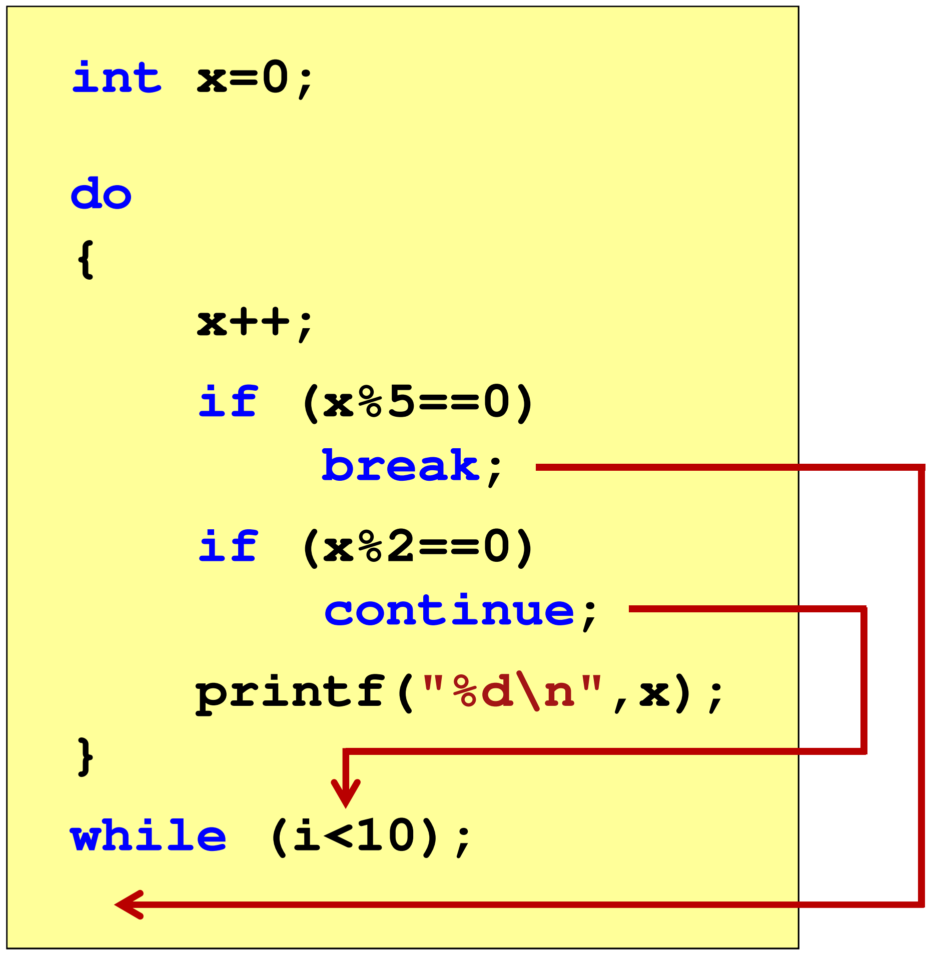
- Statement:
  - single - one statement terminated by a semicolon
  - compound - one or more statements enclosed in braces

```
int x = 10;
do
    x = x - 1;
while (x>0);
```

```
int x = 10;
do
{
    printf("%d\n",x);
    x = x - 1;
}
while (x>0);
```

The do while loop (break, continue)

- break and continue are jump statements



```c
int x=0;

do
{
    x++;
    if (x%5==0)
        break;
    if (x%2==0)
        continue;
    printf("%d\n",x);
}
while (i<10);
```

- break terminates the loop execution
- continue terminates the current iteration

Example: triangle with stars

```
Enter number (1..15): -1
Enter number (1..15): 20
Enter number (1..15): 4
*
**
***
****
```

```
#include <stdio.h>

int main(void)
{
    int i, j, n;

    do
    {
        printf("Enter number (1..15): ");
        scanf("%d",&n);
    } while (n<1 || n>15);

    for (i=1; i<=n; i++)
    {
        for (j=1; j<=i; j++)
            printf("*");
        printf("\n");
    }
    return 0;
}
```

End of workshop no. 04

Thank you for your attention!